Motivation to Understand Virtual Machines

- Why, in your opinion, we should study Virtual Machines?
- Does it have a future or is it a passing/evolving technology?
- Is it a disruptive technology?

Introduction to Virtual Machines

- Perspectives of a Machine
- Manifestation of a system
- Machine Interfaces
- System abstraction layers and virtualization
- What is system virtualization?
- Key modes of virtualization
  - Process Virtual Machines
  - System Virtual Machines
- Taxonomy of Virtual Machines

Perspectives of a Machine

- What is a Machine?
  - Hardware? (OS Perspective)
    - ISA or HAL
  - Isolated Address Space? (Process Perspective)
    - Process abstraction with APIs

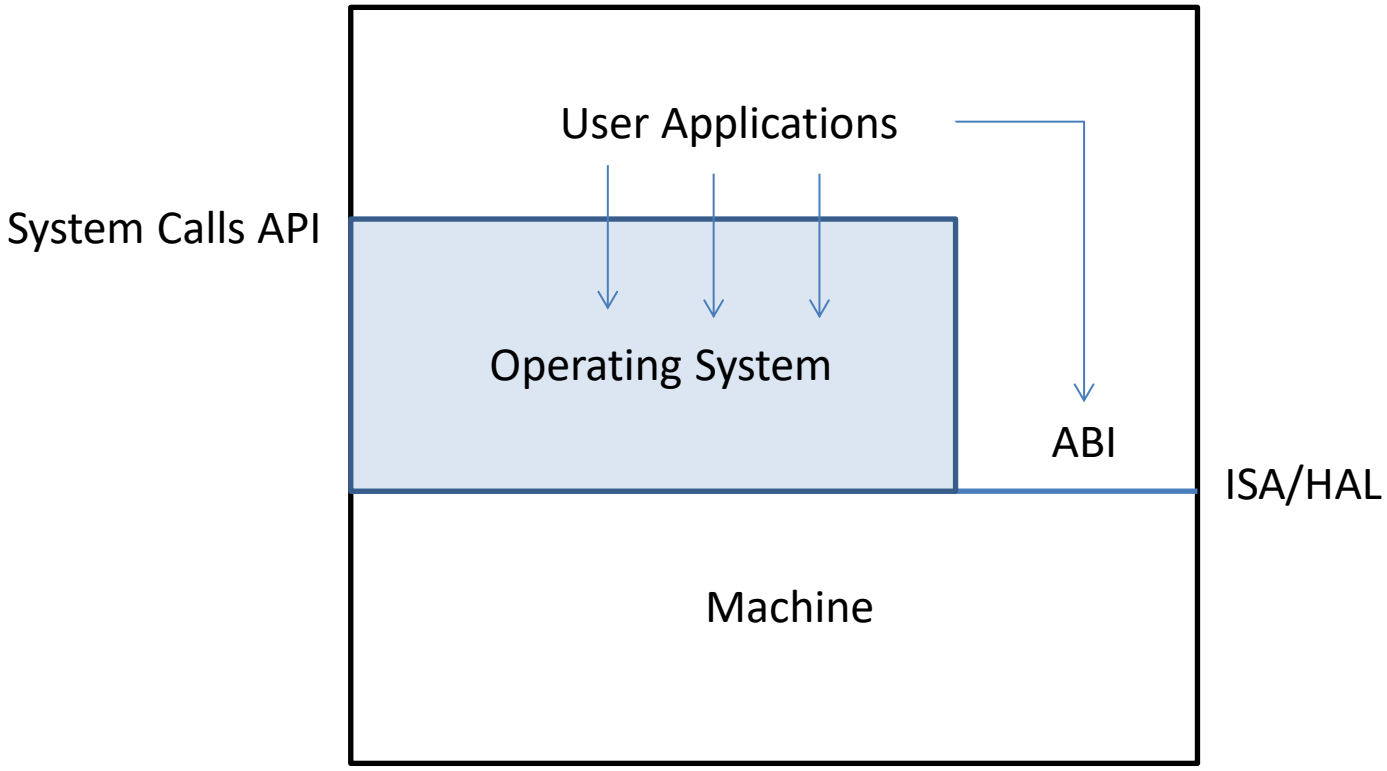
Manifestation of a System on Machine

- What is a System?
  - A Machine booted with a general purpose OS
  - Has interfaces that help users to build applications and execute them for some meaningful purpose!
- How do you build a system on a machine?
  - Machine is built to a specific ISA or HAL
  - ISA/HAL are the machine interfaces through which OS interacts with the machine.
  - OS is built for a specific ISA/HAL

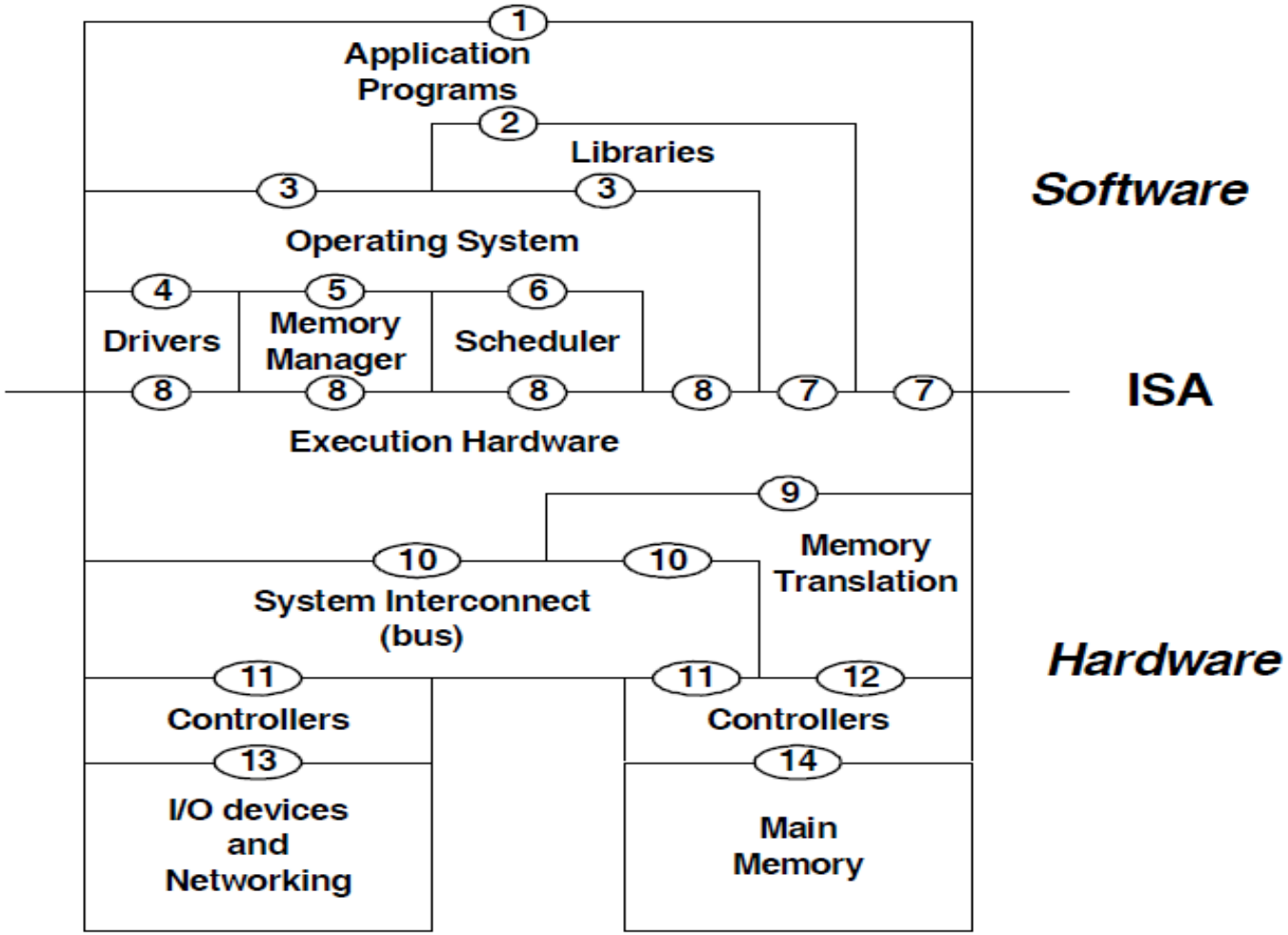
So what is a System?

- Supports full execution environment
- With multiple processes, possibly from multiple users.
- All processes share the common file system and I/O resources.
- Intrinsic assumption – all system resources are under the sole control of the OS.

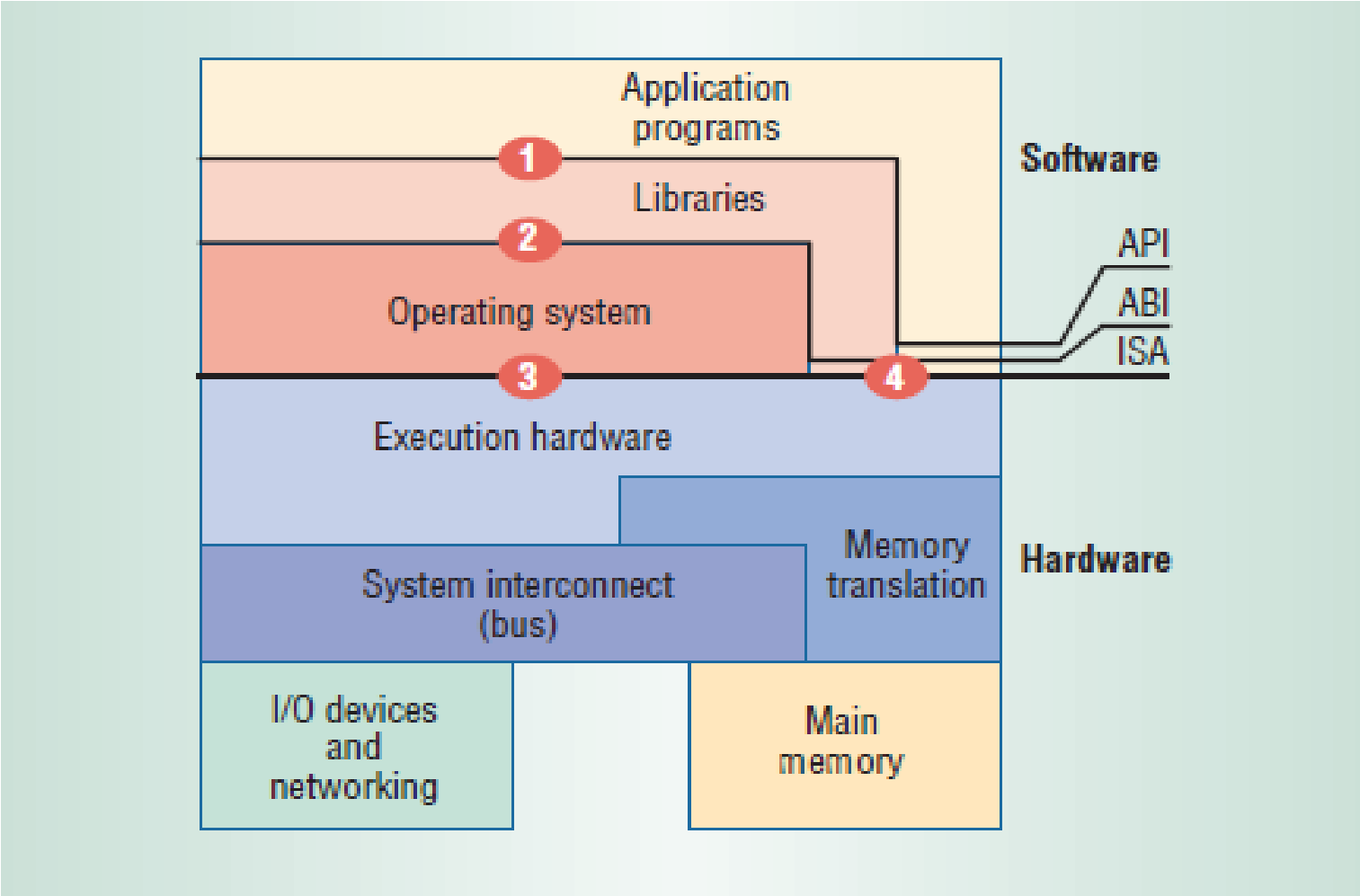
Machine Interfaces



Machine Interfaces contd.

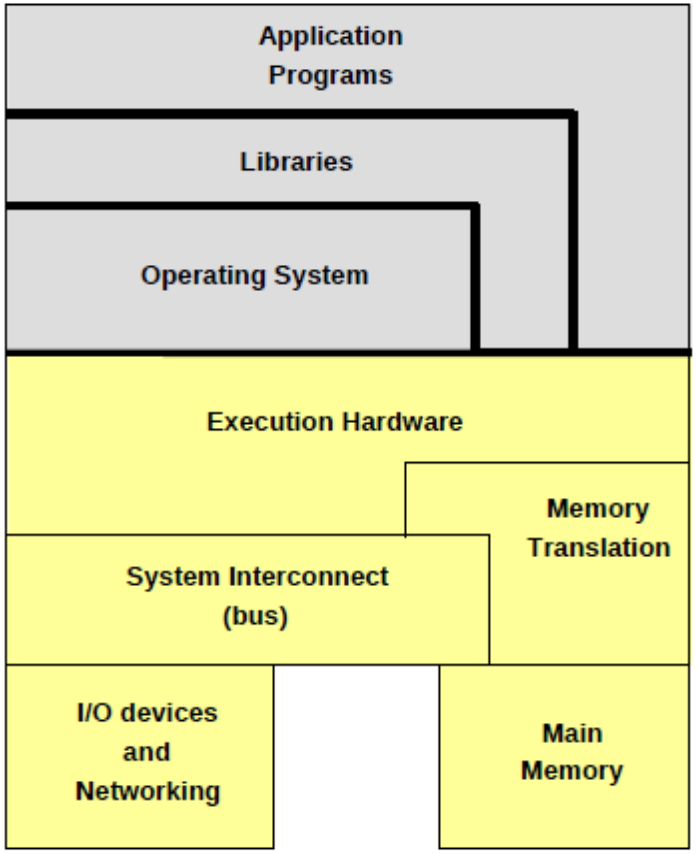


System Abstraction Layers



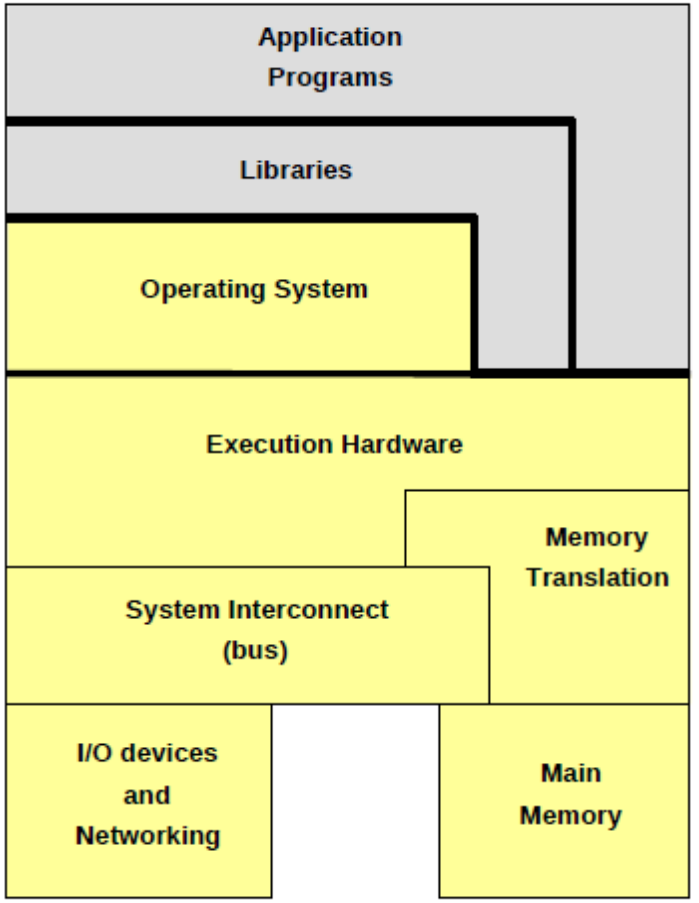
HAL/ISA Interface

- Hardware Abstraction Layer (HAL) & ISA form the basic interface between hardware and any software.
- Mostly used by OS kernel programmers
- Accessible to users through
  - User ISA + OS Systems Calls via the libraries and APIs
- System virtual machine realization



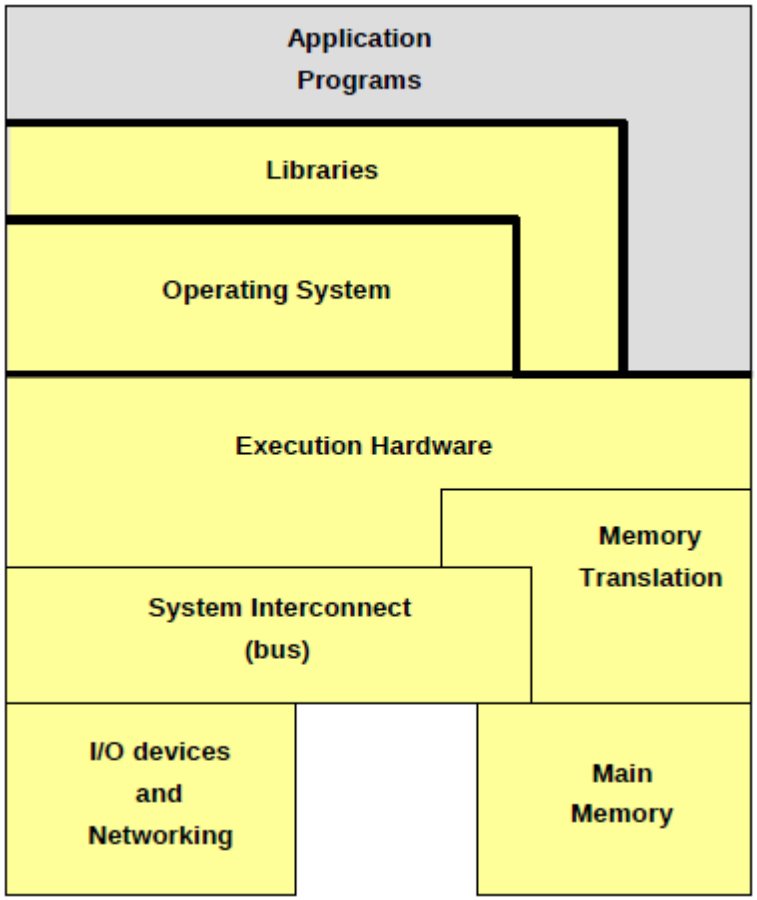
ABI Interface

- The Application Binary Interface (ABI)
  - User ISA + OS-System Calls
  - Used by compiler writers
- Platforms supporting common ABI
  - Application execution without recompilation
- Process Virtual Machine realization



API Interface

- Application Programming Interface (API)
  - User ISA + Library calls
  - Used by application programmers
- Realization of HLL based virtual machines

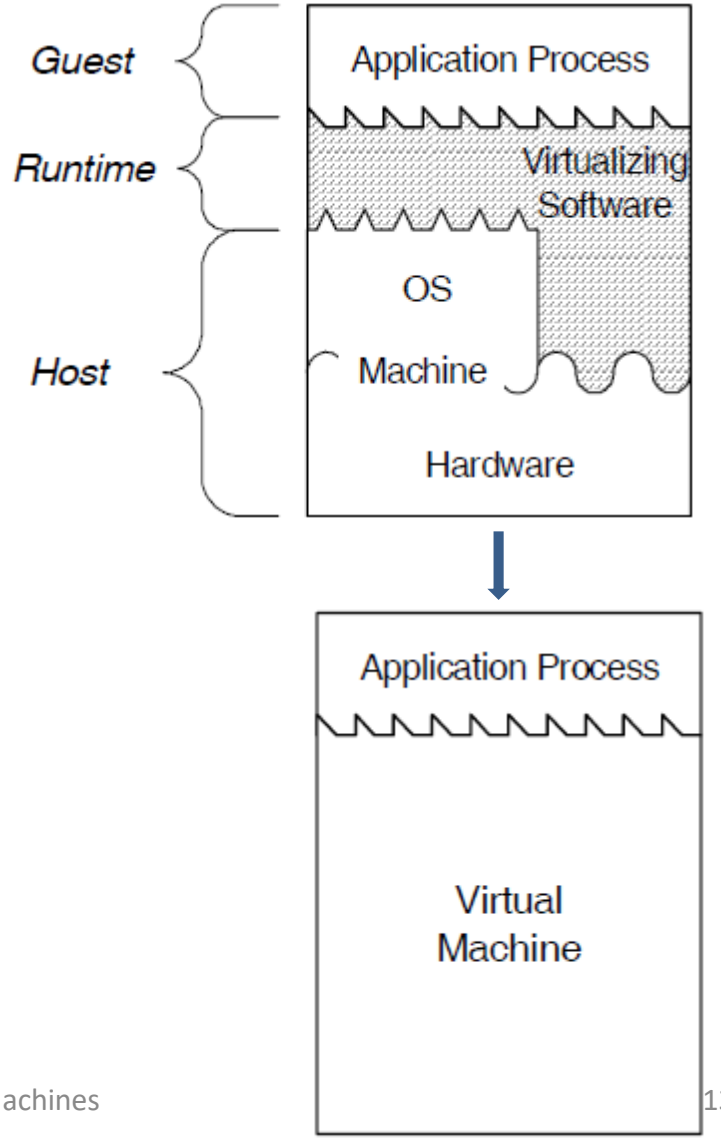


Virtual Machine?

- Evolved from both OS and Process perspectives
  - VMs as process instances on a HOST OS form the Process Virtual Machines
  - VMs as independent systems on virtual resources form System Virtual Machines

Process Virtual Machine

- A process instance of virtual machine is created over a Host-OS
- The virtualizing software resides in the Host-OS and emulates the User - ISA and OS system- calls.
- Provides the applications a virtual ABI environment using the Host-OS system call interface and the User-ISA.

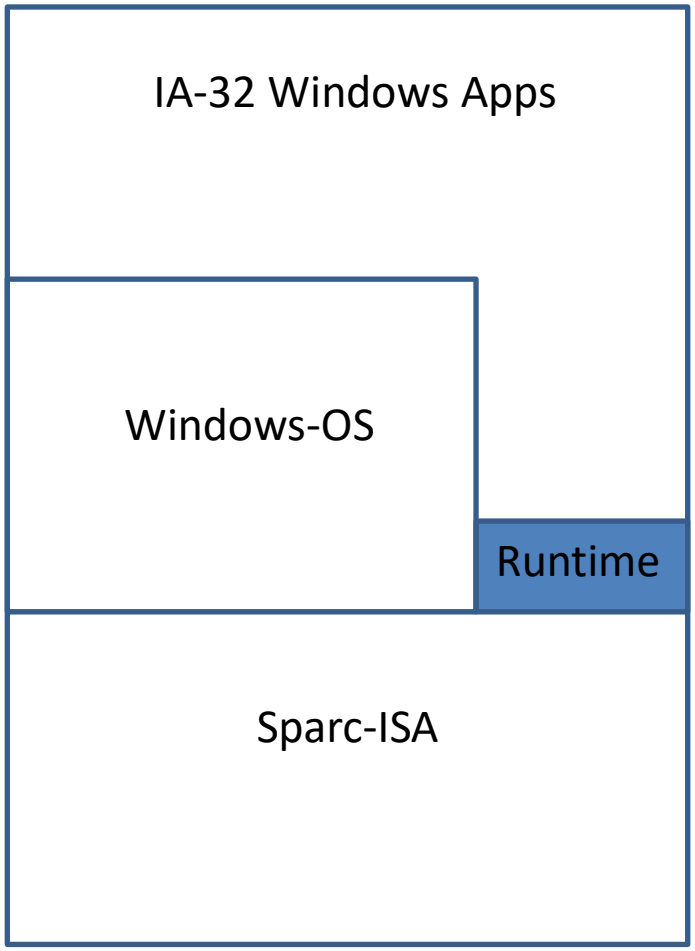


Examples of Process VMs

- Multiprogramming in general purpose OS.
  - Each application is executed within the context of a process
    - Independent, isolated address space
    - Access to system resources through OS abstractions like files, sockets, xterm, etc.
  - Each process is a replicated process-level VM that allows concurrent application execution
  - Every application running inside a process context is built for the OS environment provided by the system
- Containers also fall in this category of Process VMs

Examples of Process VMs

- Different-ABI support for process
  - An application built for a different ABI can execute within a process
  - The Host-OS provides for this feature using a technique called instruction emulation.
  - Emulation is achieved using interpretation or binary translation.
  - Binary Translators and Optimizers convert blocks of source instructions to target instructions, possibly with additional code optimization for efficient execution.
  - Same-ISA binary optimizers use the technique of dynamic code optimization with the aim of efficient execution.

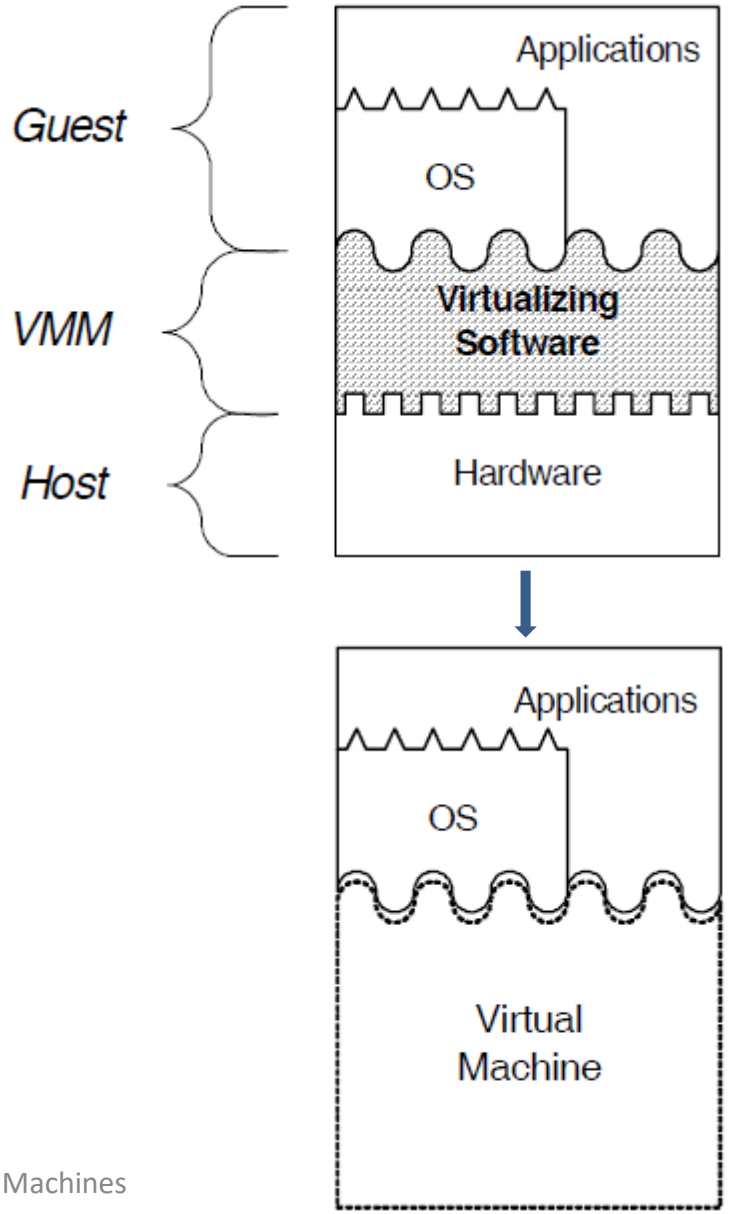


Examples of Process VMs

- High-Level Language VMs
  - Applications are built to a virtual ISA.
  - Every system contains a virtual machine that is capable of executing the virtual ISA.
  - Aim of this virtualization is software portability.
  - Common examples that use this mode of virtualization are Sun Microsystems Java VM and Microsoft CLI (Common Language Infrastructure)
  - The HLL VMs enable the building of applications into bytecode sequences.
  - The runtime consists of a set of standard libraries that use emulation techniques to convert the bytecodes into an execution sequence for a particular system.

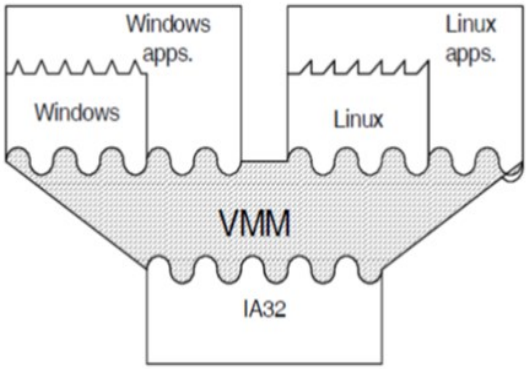
System Virtual Machine

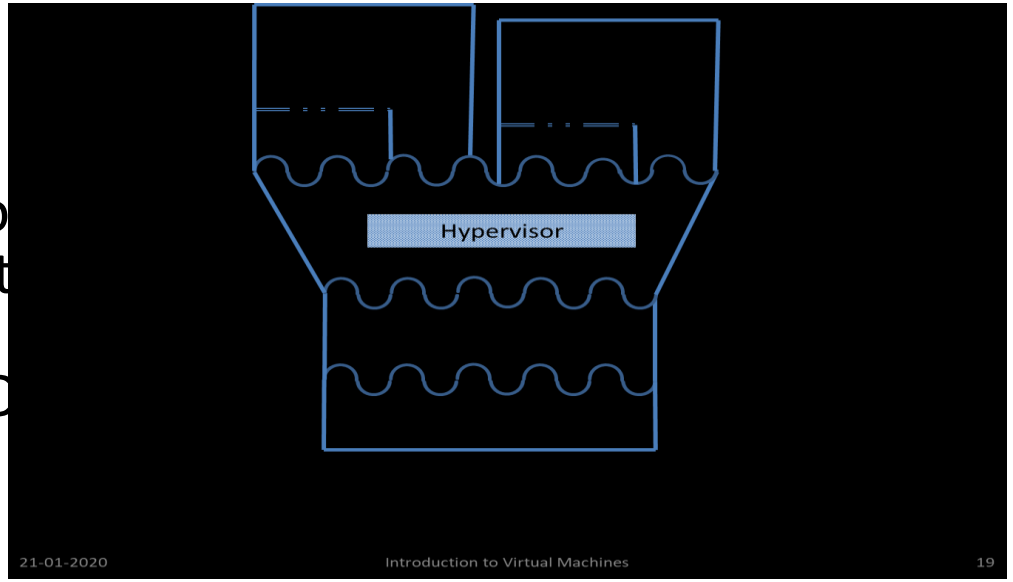
- An isolated instance of a system with replication.
- Single set of hardware resources is divided among multiple GuestOSs.
- Typically implemented as a combination of real machine and virtualizing software.
- Virtual machine may have resources different from the real machine in quantity or type or both.



Examples of System Virtual Machines

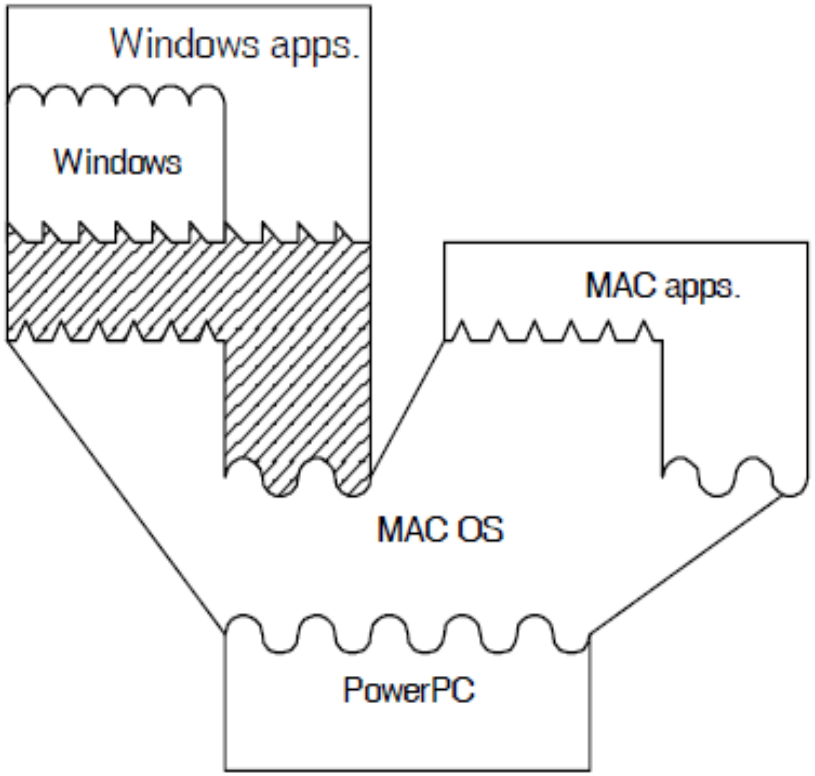
- Hypervisor based:
  - The hypervisor resides on the bare metal hardware and controls VMs' GuestOS access to the machine.
- Hosted OS based:
  - The hypervisor resides inside the Host-OS and uses features of the Ho OS for carrying out cert privileged functions required by the GuestO





Examples of System Virtual Machines

- Whole system VMs:
  - A host system running on a certain ISA can support VMs running GuestOS built for the same ISA or different ISA.
  - The VM software, GuestOS and the application are akin to a single large application built on the HostOS (i.e. use only the user ISA).
  - Example is Virtual PC

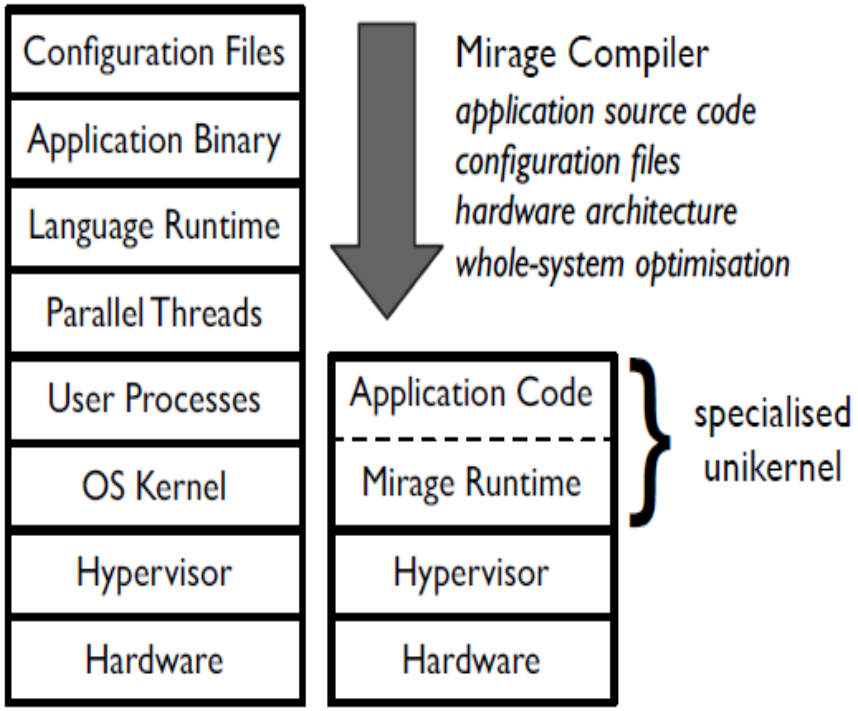


Examples of System Virtual Machines

- Co-designed VMs (Processor virtualization)
  - Targeted towards power or performance optimization.
  - Host ISA is exported as known or standard ISA which has wide OS and application base.
  - Host system implements the supported ISAs using efficient dynamic binary translation mechanisms.
  - Transmeta Crusoe and IBM AS/400 systems.

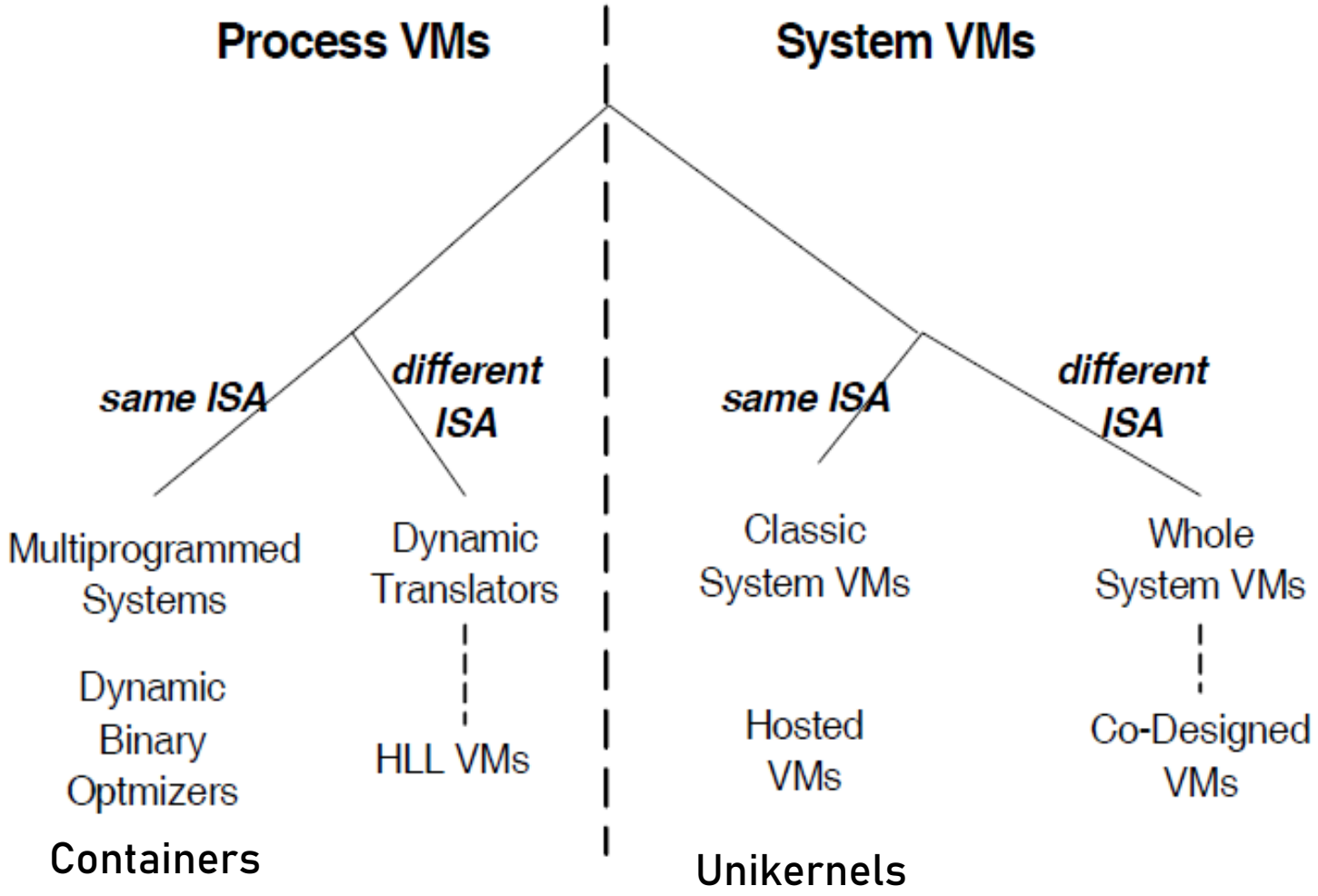
Mirage Unikernel

- Unikernels exploit the idea that individual VMs are used to provision a single application service on a general purpose OS.
- In Unikernels the OS associated functionality is captured through library OS exokernel concept and it directly interacts with the underlying hardware to execute on a system.
- Unikernels provide isolation of VMs and performance of bare-metal.



Unikernel system abstraction layers as compared to application services deployed over general purpose OS residing in a VM.

Taxonomy of VMs



Summary

- Perspectives of a machine and manifestation of the system
- System Abstraction Layers and Virtualization
- Process and system virtual machines
- Classification of virtual machines